A High-Performance Global Named Information System

Sape Mullender, Jim McKie, Jeff Napper, Jan Sacha

Bell Labs

1. Named Information

Information has always been named, but the naming mechanism usually involved a *location*, for instance, the domain name of the server publishing the information. There is a lot of research a the moment in the area of Named-Information systems: Information-Centric Networks (ICN), Content-Centric Networks (CCN), Named-Data Networks (NDN), as well is into Content-Delivery Networks (CDNs) that looks into using the name of an object to map it to a nearby cached copy, plus, of course, the algorithms for deciding just what information to cache and where to do that.

The research mentioned could be described as research that attempts to integrate information *communication* and *storage*. In this position paper, we plead to integrate this with information *creation* or *generation* as well and to apply it not only to distributed systems in general but to exascale systems in particular.

Searching for web pages with certain keywords in them can only be done by servers in the network. But information that is expensive to generate (a weather forecast, for example), tends to be generated once, proactively, and then to be distributed by CDNs. In other words, we either have distributed applications for generating information, *or* CDNs for delivering pre-generated information. The integration of the two, for information that can either be delivered from storage or generated on demand (depending on the cost of fetching it vs. the cost of generating it) has scarcely been investigated.

2. Information Systems

Imagine a global system in which *naming* the information triggers the mechanisms that deliver it. Such mechanisms include information generation, caching, transportation, as well as the management of privacy and integrity. Not all information is accessible to anyone and not all information can be provided by just anyone. Moreover, there must be mechanisms to manage timely delivery, consistency of replicated information (i.e., change management) and more.

We conjecture a new protocol stack for named information management in which the waist of the protocol hourglass addresses (1) the name space, (2) operations on information (create/delete, read/update) and (3) access control (who can do what). Below this layer are the mechanisms for authentication, transport, data encryption, information location and more. Above the waist are the ways in which information is interpreted, how and where information is replicated or cached, and how fault-tolerance parameters are managed.

Basically, the waist of the hourglass is a protocol that describes the exchanges between a client seeking to obtain or update information managed by a server. A *cache* can be viewed as an agent presenting itself to the client as a server and to another cache, a file system, or a server process as a client. The protocol manages the information exchanges as well as the mechanisms for maintaining consistency, performing access control and managing timely data delivery.

The protocol is reminiscent of the protocols used for clients accessing remote file systems and this is not entirely accidental: file I/O is something we know how to do and it's been used as the driving model in Unix: processes look like files (via *pipes*), devices look like files and some services are presented as files (e.g., `/proc`). It's a familiar model that we know how to use. Moreover, it's already part and parcel of most of the applications we run.

Adapting it to high-performance environments, to transatlantic communication, to interactive communication and to dynamically generated content is a challenge that we address using two important protocol components:

1. A **session** is established between communicating entities. It captures the *state* shared by the participants: identity/authenticity of the participants, context for naming information (keeping the names that need to be exchanged short) operations in progress. A session can be established once and used for the duration of a running application or a client-server relationship.

2. A **transport** connection is established for exchanging names and information between participants. Each connection belongs to a *session*; sessions can have zero, one of many connections at any given moment.

Disruptions in communication due to address changes can be dealt with at session level: a new connection is established in the existing session and communication state can be recovered from the session. Authentication handshakes on new connections are usually not needed because pre-established keys can be reused.

The mechanism used to communicate in a connection will depend on the proximity of the participants: two cores on the same processor might use shared memory, while processes on either side of the Atlantic will use IP. The session allows connections to be established in the most efficient manner each time a participant migrates — or is restarted from a checkpoint after a failure.

The information-oriented network's equivalent of a conventional network's data multicast is *content distribution*, giving a whole new meaning to *store-and-forward* in content-delivery systems.

The protocol at the waist of the hourglass, first and foremost, needs to be capable of getting the job done: content distribution, publish/subscribe applications, or consistent distributed data management via a network of caches and storage servers; addressing distributed or centralized services in order to make them generate or transform information; dealing with the demands of data and system security.

3. Relevance to Exascale Computing

Maturity

Many of the communication demands in exascale computing have parallels in current distributed applications and CDN systems:

Exascale systems, due to their increasing size, are becoming much more heterogeneous and data transport between cores depends on how directly those cores are connected. The separation of concerns between session and connection allows the most direct connection to be used while the session guards the longer-term concerns of error-recovery, process migration and authentication.

Exascale applications often need to share data efficiently, typically through multicast. In the emerging heterogeneous exascale architectures, multicast can no longer be done by broadcasting data on a bus or an Ethernet segment, making the multicast architecture resemble a content-distribution network. We consider it a feasible challenge to model information multicast in exascale applications on the lines for content distribution without significant loss of efficiency.

Uniqueness and Applicability

Our approach, obviously, is not unique to exascale systems, but exascale systems are explicitly targeted in the realization that meeting the extreme performance demands of exascale systems will benefit other systems as well.

Novelty

Our focus on *named* information allows the system to be used *as if* it were a file system. Naturally, many aspects of the realization are not like actual file systems at all, but the opportunity presents itself to make a coupling between, on the one hand, exascale applications running on exascale operating systems and, on the other, conventional operating systems with conventional file systems (and conventional synthetic files for I/O and control).

Experiments are ongoing to model running processes as named information, providing methods for monitoring, controlling, debugging, checkpointing and migrating processes. This is a new and very scalable approach to managing applications consisting of thousands of processes.

Effort

A project at Bell Labs has been underway for about a year in which the current approach is being evaluated for embedded and conventional operating systems. Making the adaptations for HPC systems and exascale systems is a project that is expected to take three years and between 8 and 20 person years.